REMARKS

Claims 1-10 are pending and all have been rejected under either 35 U.S.C. § 102. In response, Applicants are not amending any claim, but are adding claims 11-16, and respectfully submits that all pending claims 1-10 and added claims 11-16 present subject matter that is patentable over the prior art of record, and, in view of the above amendments and following remarks, requests that the Examiner reconsider the application.

REJECTIONS UNDER 35 U.S.C. § 102(b) – Copperman (PMW)

In paragraphs 2 and 3 of the Office Action, the Examiner rejected claims 1-10 under 35 U.S.C. § 102 (b) as being anticipated by *Copperman et al*, "Poor Man's Watchpoint," referred to as *PMW*.

Regarding claim 1, the Examiner asserted that *PMW* teaches:

- a method for allowing debugging capability in code instrumentation ("supplying an interface to the instrumentation in the debugger . . ." on page 37)
- that takes a block of original code to produce a block of instrumented code ("replacing each store and/or load instructions with an inline check or call to a function . . ." on page 37)
- providing an instrumentation breakpoint in the block of original code ("a call to _do_watch is patched into the executable prior to each memory access." on page 38)
- generating the block of instrumented code ("replacing each store and/or load instructions with an inline check or call to a function . . ." on page 37)
- running the block of instrumented code until a debugging breakpoint is reached ("replacing each store and/or load instructions with an inline check or call to a function that gives control to the debugger if the accessed location is being watched . . ." on page 37)
- performing debugging functions on the block of instrumented code ("On receiving a watchpoint command, the debugger has to add an entry to the watch table . . ." on page 40 as claimed.

Applicants respectfully submit that Applicants' claim 1 is patentably distinguished from *PMW* for at least several reasons. In general, Applicants' claim 1 recites *a block of original code* and *a block of instrumented code,* and the

programming flow between the block of original code and the block of instrumented code, such as providing an instrumentation breakpoint *in the block of original code* and *generating the block of instrumented code when the instrumentation breakpoint is reached.* Because the instrumentation breakpoint is included in the block of original code, it is implied that the block of original code is invoked for the instrumentation breakpoint to be reached, then the block of instrumented code is generated. However, *PMW* discloses the instrumented code "[b]y instrumenting the program code to check memory accesses" (Abstract, page 37), but does not teach, suggest, or make obvious providing an instrumentation breakpoint in the block of original code or generating the block of instrumented code when the instrumentation breakpoint is reached as in Applicants' claim 1. *PMW*'s debugger sees only the instrumented code, and there is no teaching of programming flow between the original code and the instrumented code. For example, *PMW*'s page 37, near the end, recites "code patching – replacing each store and/or load instructions with an inline check;" *PMW*'s page 38, near the middle of the page, recites "[a] call to _do_watch is patched into the executable prior to each memory access," *PMW*'s page 39, in the paragraph starting with "[t]he first time," recites "[w]hen the patched program was run under debugger control . . ." etc., but there is no discussion regarding debugging in conjunction with the original block of code.

Applicants' claim 1 recites *an instrumentation breakpoint* and a *debugging breakpoint* while the Examiner corresponded Applicants' instrumentation breakpoint to *PMW*'s call to _do_watch, but failed to correspond Applicant's debugging breakpoint to any other breakpoint in *PMW*.

Applicants' instrumentation breakpoint is provided *in the block of original code*, and if *PMW*'s call to _do_watch corresponded to Applicants' instrumentation

breakpoint, then this call to _do_watch must be provided in the block of original code. However, *PMW*'s call to _do_watch is provided in the instrumented code.

Applicants' claim 1 recites "generating the block of instrumented code *when the instrumentation breakpoint is reached*" (emphasis added). Even though *PMW* discloses "replacing each store and/or load instructions with an inline check . . ." (page 37, near the end), *PMW* fails to teach such replacement occurs *when the instrumentation breakpoint is reached.* For the sake of argument, if *PMW*'s call to _do_watch corresponded to Applicants' instrumentation breakpoint, then, to be parallel with Applicants' claim 1, the replacement of the store and/or load instructions with an inline check or call to a function, which the Examiner corresponded to Applicants' block of instrumented code, *must be generated when a call to _do_watch is reached.* However, there is no such teaching in *PMW.*

Because claim 1 recites limitations patentably distinguished from *PMW*, claim 1 is patentable.

Claims 2-5 depend directly or indirectly from claim 1 and are therefore patentable for at least the same reasons as claim 1. Claims 2-5 are also patentable for their additional limitations as appropriate. For example, claims 2 and 3 recite "the step of providing comprises the step of replacing a fist instruction in the block of original code . . .," which is not taught, suggested, or made obvious by *PMW.* The Examiner asserted that a call to _do_watch is patched into the executable prior to each memory access, and "[t]he patch replaces the code section before the memory access block of code." However, Applicants respectfully submit that there is no teaching of such replacement in *PMW.* Code may be patched into another code section without replacing other code or instructions.

Independent claim 6 recites limitations corresponding to claim 1, and is patentable for at least the same reasons as claim 1.

Claims 7-10 depend directly or indirectly from claim 6 and are therefore patentable for at least the same reasons as claim 6. Claims 7-10 are also patentable for their additional limitations as appropriate.

Because dependent claims 2-5 and 7-10 are patentable for at least the same reasons as their corresponding independent claims 1 or 6, Applicants are not responding to some of the Examiner's assertion regarding limitations of the dependent claims, but wish to reserve the rights to late respond to these assertions as appropriate.

NEW CLAIMS

Applicants are adding claims 11-16, which depend from claims 1 or 6, and respectfully submit that these added claims are patentable for at least the same reasons as claims 1 or 6 from which these dependent claims 11-16 depend. These added claims are also patentable for their own limitations.

Applicants further submit that all limitations in the added claims are supported in the Specification and therefore no new matter is being added.

SUMMARY

In conclusion, Applicants respectfully submit that pending claims 1-10 and added claims 11-16 clearly present subject matter that is patentable over the prior art of record, and therefore request that the Examiner withdraw the rejections of the pending claims, consider the added claims, and pass the application to issue. If the Examiner has questions regarding this case, the Examiner is invited to contact Applicants' undersigned attorney.

Respectfully submitted,

Robert Hundt et al

Date: 3/26/04 By: _____

Tuan V. Ngo, Reg. No. 44,259
IP Administration
Legal Department, M/S 35
Hewlett-Packard Company
P. O. Box 272400
Fort Collins, CO 80527-2400
Phone (408) 447-8133
Fax (408) 447-0854